Practice Problems for Quiz 1

Scope: Chapters 1, 2, 3, 4, 5, and 7 in Learn You a Haskell for Great Good.

Some standard Haskell functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>Bool -&gt; Bool</td>
</tr>
<tr>
<td>(&amp;&amp;),(</td>
<td></td>
</tr>
<tr>
<td>(+++)</td>
<td>[a] -&gt; [a] -&gt; [a]</td>
</tr>
<tr>
<td>(:)</td>
<td>a -&gt; [a] -&gt; [a]</td>
</tr>
<tr>
<td>concat</td>
<td>[[a]] -&gt; [a]</td>
</tr>
<tr>
<td>concatMap</td>
<td>(a -&gt; [b]) -&gt; [a] -&gt; [b]</td>
</tr>
<tr>
<td>delete</td>
<td>(Eq a) =&gt; a -&gt; [a] -&gt; [a]</td>
</tr>
<tr>
<td>drop, take</td>
<td>Int -&gt; [a] -&gt; [a]</td>
</tr>
<tr>
<td>elem</td>
<td>(Eq a) =&gt; a -&gt; [a] -&gt; Bool</td>
</tr>
<tr>
<td>filter</td>
<td>(a -&gt; Bool) -&gt; [a] -&gt; [a]</td>
</tr>
<tr>
<td>foldl</td>
<td>(a -&gt; b -&gt; a) -&gt; a -&gt; [b] -&gt; a</td>
</tr>
<tr>
<td>foldr</td>
<td>(a -&gt; b -&gt; b) -&gt; b -&gt; [a] -&gt; b</td>
</tr>
<tr>
<td>fst</td>
<td>(a,b) -&gt; a</td>
</tr>
<tr>
<td>snd</td>
<td>(a,b) -&gt; b</td>
</tr>
<tr>
<td>head, last</td>
<td>[a] -&gt; a</td>
</tr>
<tr>
<td>init, tail, reverse</td>
<td>[a] -&gt; [a]</td>
</tr>
<tr>
<td>isLower, isUpper</td>
<td>Char -&gt; Bool</td>
</tr>
<tr>
<td>length</td>
<td>[a] -&gt; Int</td>
</tr>
<tr>
<td>map</td>
<td>(a -&gt; b) -&gt; [a] -&gt; [b]</td>
</tr>
<tr>
<td>null</td>
<td>a -&gt; [a] -&gt; Bool</td>
</tr>
<tr>
<td>product, sum</td>
<td>(Num a) =&gt; [a] -&gt; a</td>
</tr>
<tr>
<td>zip</td>
<td>[a] -&gt; [b] -&gt; [(a,b)]</td>
</tr>
<tr>
<td>zipWith</td>
<td>(a -&gt; b -&gt; c) -&gt; [a] -&gt; [b] -&gt; [c]</td>
</tr>
</tbody>
</table>

Question 1 (6 points). Write a function

\[
\text{insert :: Int} \rightarrow a \rightarrow [a] \rightarrow [a]
\]

such that (insert n x ys) returns a new version of the list ys where x is inserted as the n-th element, where indexing is 0-based. (We assume \(n \geq 0\).) When \(n \geq (\text{length } ys)\), the function inserts x as the last element.

Examples:

- insert 0 42 [0..3] \(\sim [42,0,1,2,3]\)
- insert 1 42 [0..3] \(\sim [0,42,1,2,3]\)
- insert 2 42 [0..3] \(\sim [0,1,42,2,3]\)
- insert 3 42 [0..3] \(\sim [0,1,2,42,3]\)
- insert 4 42 [0..3] \(\sim [0,1,2,3,42]\)

- insert 99 42 [0..3] \(\sim [0,1,2,3,42]\)

Question 2 (6 points). Write a function

\[
\text{shortest :: [[a]]} \rightarrow [a]
\]

that given a nonempty list of lists, returns a shortest list from the input. (Break ties for shortest however you want.)

Example: \((\text{shortest } [["this","is","a","test"]]) \sim "a"

Question 3 (4 points). This problem involves the following data-type for binary trees.

\[
\text{data BinTree = Emp \mid Branch Int BinTree BinTree}
\]

Write a function

\[
\text{sqBinTree :: BinTree} \rightarrow \text{BinTree}
\]

such that \((\text{sqBinTree } t)\) returns a new version of \(t\) in which each number is replaced by its square.

Example: \((\text{sqBinTree (Branch 2 (Branch 3 Emp Emp) (Branch 5 Emp Emp)))}) \sim \text{Branch 4 (Branch 9 Emp Emp) (Branch 25 Emp Emp)}

Question 4 (4 points). This problem involves the following data-type for multi-way trees.

\[
\text{data Multi = Fork Int [Multi]} \rightarrow \text{Fork Int [Multi]}
\]

Write a function

\[
\text{sqMulti :: Multi} \rightarrow \text{Multi}
\]

such that \((\text{sqMulti } t)\) returns a new version of \(t\) in which each number is replaced by its square.

Example: \((\text{sqMulti (Fork 2 [Fork 3 [Fork 5 [],Fork 7 []]])}) \sim \text{Fork 4 [Fork 9 [Fork 25 [],Fork 49 []]]}

Problem 1, Possible Answer

\[
\text{insert}, \text{insert'} :: \text{Int} \to \text{a} \to \text{[a]} \to \text{[a]}
\]

\[
\text{insert} \ n \ x \ ws = xs ++ x:ys \\
\text{where} \ (xs,ys) = \text{splitAt} \ n \ ws
\]

-- or --

\[
\text{insert'} \ 0 \ x \ ws = x:ws \\
\text{insert'} \ n \ x \ [] = [x] \\
\text{insert'} \ n \ x \ (w:ws) = \text{insert} \ (n-1) \ x \ ws
\]

Problem 2, Possible Answer

\[
\text{shortest}, \text{shortest'}, \text{shortest''} :: \text{[[a]]} \to \text{[a]}
\]

\[
\text{shortest} \ [xs] = xs \\
\text{shortest} \ (xs:xss) = \text{if} \ \text{length} \ xs < \text{length} \ ys \ \text{then} \ xs \ \text{else} \ ys \\
\text{where} \ ys = \text{shortest} \ xss
\]

-- or --

\[
\text{shortest'} \ (xs:xss) = \text{foldr} \ \text{cmp} \ xs \ xss \\
\text{where} \ \text{cmp} \ ys \ zs = \text{if} \ \text{length} \ ys < \text{length} \ zs \ \text{then} \ ys \ \text{else} \ zs
\]

-- or, digging way deep into the libraries, --

\[
\text{shortest''} = \text{minimumBy} \ (\text{comparing} \ \text{length})
\]

Problem 3, Possible Answer

\[
\text{sqBinTree} :: \text{BinTree} \to \text{BinTree}
\]

\[
\text{sqBinTree} \ \text{Emp} = \text{Emp} \\
\text{sqBinTree} \ \text{Branch} \ n \ tl \ tr = \text{Branch} \ (n*n) \ (\text{sqBinTree} \ tl) \ (\text{sqBinTree} \ tr)
\]

Problem 4, Possible Answer

\[
\text{sqMulti}, \text{sqMulti'} :: \text{Multi} \to \text{Multi}
\]

\[
\text{sqMulti} \ \text{Fork} \ n \ ts = \text{Fork} \ (n*n) \ (\text{map} \ \text{sqMulti} \ ts)
\]

-- or --

\[
\text{sqMulti'} \ (\text{Fork} \ n \ ts) = \text{Fork} \ (n*n) \ [\text{sqMulti'} \ t \mid t <- ts]
\]